Operations Research Forum (2026) 7:26
https://doi.org/10.1007/s43069-026-00610-9

RESEARCH

Genetic Algorithm-Based Shape Parameter Tuning for Radial Basis Function Interpolation

Roberto Cavoretto[1] · Alessandra De Rossi[1] · Sandro Lancellotti[1] · Domenico Mezzanotte[2,3]

Received: 17 September 2025 / Accepted: 22 January 2026

Abstract
In this study, we address the non-trivial problem of determining the optimal shape parameter in radial basis function interpolation. We propose the use of a genetic algorithm, an optimization technique inspired by evolutionary selection, to identify this parameter. Numerical experiments are presented to evaluate the effectiveness of this approach, with direct comparisons to the leave-one-out cross-validation method, thereby highlighting its computational efficiency.

Keywords Radial basis functions · Kernel-based interpolation · Shape parameter · Genetic algorithm · Scattered data

Mathematics Subject Classification (2010) 65D12 · 65D15 · 68W50

Roberto Cavoretto, Alessandra De Rossi, Sandro Lancellotti, and Domenico Mezzanotte contributed equally to this work.

✉ Domenico Mezzanotte
domenico.mezzanotte@unibas.it

Roberto Cavoretto
roberto.cavoretto@unito.it

Alessandra De Rossi
alessandra.derossi@unito.it

Sandro Lancellotti
sandro.lancellotti@unito.it

[1] Department of Mathematics "Giuseppe Peano", University of Turin, Via Carlo Alberto 10, 10123 Turin, Italy

[2] Department of Basic and Applied Sciences, University of Basilicata, Via dell'Ateneo Lucano 10, 85100 Potenza, Italy

[3] Istituto per le Applicazioni del Calcolo "Mauro Picone", Naples Branch, C.N.R. National Research Council of Italy, Via Pietro Castellino 111, 80131 Naples, Italy

Published online: 16 February 2026

1 Introduction

Over the last decades in the literature, there has been an increasing interest in multivariate interpolation, numerical cubature, and numerical methods for integral and differential equations, in particular when the available data are scattered. In this context, many meshless methods have been developed in various forms, for instance, those that rely on radial basis functions (RBFs) [2, 27]. Some RBFs are scaled by a shape parameter, while others are shape-parameter-free. The task of searching for the optimal shape parameter is non-trivial. Indeed, the shape parameter can improve the accuracy of the interpolation, but at the same time can also lead to an ill-conditioned interpolation matrix, and hence to numerical instabilities. The optimal shape parameter should take these two aspects into account and balance them. The challenge is further compounded by the fact that the choice of the optimal shape parameter often depends on the specific characteristics of the problem at hand, such as the distribution of the data points and the nature of the function being approximated. In the literature, there are several different techniques that have been proposed to find it (see, e.g., [11, Chapter 17] or [8, 13, 18]). Some of the approaches useful to search for the optimal shape parameter in RBF interpolation are based on maximum likelihood estimation and Bayesian/Lipschitz optimization (see, e.g., [3, 4, 16, 21], and [12, Chapter 14] for an overview). Each of these techniques leads to different advantages and limitations, highlighting the diverse strategies that researchers have developed in the years to address the challenging nature of this optimization problem.

The genetic algorithm (GA) is a method for solving optimization problems that is based on natural selection. GAs belong to the class of evolutionary algorithms and make use of mechanisms inspired by biological evolution, such as reproduction, mutation, and selection. In this context, candidate solutions to the optimization problem become individuals in a population, and the fitness function determines the quality of these solutions. The population evolves towards better solutions through an iterative process that stops when a maximum number of generations has been produced or the population reaches a satisfactory fitness level. In this work, the GA is proposed as a possible alternative to cross-validation (CV) algorithms such as the leave-one-out cross-validation (LOOCV) technique [19]. The key idea of this procedure is to minimize a cost function based on errors from partial fits to the data. In this procedure, one point is removed from the dataset to construct a training set used to build a partial interpolation. The omitted point then serves as a validation set to compute the error. This process is repeated for each point in the dataset, producing a vector of errors. Hence, while effective, LOOCV can be computationally expensive, especially for large datasets, as it requires solving the interpolation problem repeatedly for each omitted point. Despite this, its simplicity and reliability have made it a benchmark method in this field. In contrast, the GA offers a valuable alternative due to reduced computational overhead, making it particularly suitable for large-scale problems or scenarios where computational resources are limited.

The outline of the paper is as follows. In Section 2, we report some preliminaries on multivariate RBF interpolation and recall CV-based algorithms. The GA is described in Section 3, while in Section 4, the proposed algorithms are explained in detail. Section 5 focuses on showing some numerical experiments, also comparing the algorithms' performance. Finally, Section 6 concludes the paper.

2 Preliminaries

2.1 Radial Basis Functions

Suppose to have in a compact domain $\Omega \subset \mathbb{R}^d$ a set of scattered data points $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\} \subset \Omega$ and a corresponding set of function values $F = \{f_1, \ldots, f_N\} \subset \mathbb{R}$, which represents samples of some unknown function $f : \Omega \to \mathbb{R}$.

Moreover, consider a radial basis function (RBF) $\phi : \mathbb{R}_+ \to \mathbb{R}$ depending on a shape parameter $\varepsilon > 0$ and strictly conditionally positive definite (SCPD) of order m. Denoting by $\|\cdot\|_2$ the Euclidean norm, we set the notation

$$\phi_{\varepsilon,i}(\boldsymbol{x}) := \phi(\varepsilon\|\boldsymbol{x} - \boldsymbol{x}_i\|_2) = \phi(\varepsilon r), \quad r \geq 0,$$

and construct a unique interpolating function $q : \Omega \to \mathbb{R}$ such that

$$q(f, \boldsymbol{x}) = \sum_{i=1}^{N} c_i \phi_{\varepsilon,i}(\boldsymbol{x}) + \sum_{i=N+1}^{N+M} c_i \pi_{i-N}(\boldsymbol{x}), \tag{1}$$

where $\{\pi_k\}_{k=1}^{M}$ form a basis for the $M = \binom{m-1+d}{m-1}$-dimensional space $\mathbb{P}^d_{m-1}$ of d-variate polynomials of total degree $\leq m-1$. The coefficients $c_1, \ldots, c_{N+M}$ are determined by imposing the interpolation conditions

$$q(f, \boldsymbol{x}_i) = f_i, \quad i = 1, \ldots, N.$$

This results in a system of N linear equations with $N + M$ unknowns, so the additional conditions

$$\sum_{i=1}^{N} c_i \pi_k(\boldsymbol{x}_i) = 0, \quad k = 1, \ldots, M,$$

are imposed to ensure a unique solution.

It is known that if ϕ is SCPD of order 0, it is strictly positive definite (SPD), allowing the polynomial term in Eq. 1 to be omitted. In general, solving the interpolation problem for an SCPD function of order m yields a symmetric linear system

$$\mathcal{A}_\varepsilon \boldsymbol{c} = \boldsymbol{d}, \tag{2}$$

with

$$\mathcal{A}_\varepsilon = \begin{bmatrix} A_\varepsilon & P \\ P^T & O \end{bmatrix}, \quad \boldsymbol{d} = \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{0} \end{bmatrix}.$$

The interpolation matrix $\mathcal{A}_\varepsilon$ consists of elements

$$a_{ij} = \phi(\varepsilon\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2), \quad P_{ik} = \pi_k(\boldsymbol{x}_i), \quad i, j = 1, \ldots, N, \; k = 1, \ldots, M,$$

O being an $M \times M$ zero matrix. The vector $\boldsymbol{c} = (c_1, \ldots, c_{N+M})^T$, $\boldsymbol{f} = (f_1, \ldots, f_N)^T$, and $\boldsymbol{0}$ is the null vector of length M.

Note that when ϕ is SPD, the linear system in Eq. 2 reduces to

$$A_\varepsilon \overline{\boldsymbol{c}} = \boldsymbol{f}, \tag{3}$$

where $\overline{\boldsymbol{c}} = (c_1, \ldots, c_N)^T$. Recalling that every SCPD kernel has an associated normalized PD kernel [20], from now on, we confine our treatise to the case of SPD kernels.

2.2 Cross-validation Techniques

CV methods are techniques for estimating the optimal RBF shape parameter ε. The key idea is to minimize a cost function based on errors from partial fits to the data. More in detail, given a dataset with N data points, in the k-fold CV, the dataset is divided into k disjoint subsets (possibly of equal size), where $k \leq N$. Iteratively, $k \in \mathbb{N}$ models are constructed using $k-1$ subsets as training data, and their performance is assessed on the remaining subset, which serves as the validation fold. An alternative CV approach is known as leave-p-out cross-validation (LpOCV) [10], where $p \in \mathbb{N}$ and $p < N$, and all possible combinations of p elements from the dataset are used as validation sets. Since this method is computationally intensive in many cases, k-fold CV is typically preferred. In this work, we use LpOCV as a notation to indicate k-fold CV with $k \approx N/p$, and we recall it as an extended version of Rippa's algorithm [17] that includes also the original Rippa's algorithm [19] as a particular case. Indeed, when $k = N$ in k-fold CV, this corresponds to LpOCV with $p = 1$, commonly referred to as LOOCV, as each validation fold consists of a single data point. This LOOCV approach effectively performs an exact N-fold CV and has been extensively utilized in the literature.

Consider one of the k folds. We define a vector $\boldsymbol{p} = (p_1, \ldots, p_v)^T$ containing distinct validation indices $p_j \in \{1, \ldots, N\}$, where $v \in \mathbb{N}$ and $v < N$. The dataset is then divided into a training set $\mathcal{T}$ of $N - v$ points $(\boldsymbol{x}_j, f_j)$ with indices $j \notin \boldsymbol{p}$, and a validation set $\mathcal{V}$, containing the remaining v points $(\boldsymbol{x}_{p_j}, f_{p_j})$ for $j = 1, \ldots, v$.

For a fixed ε, the partial RBF interpolant on $\mathcal{T}$ is defined as follows:

$$q^{[\boldsymbol{p}]}(f, \boldsymbol{x}) = \sum_{i=1,\ i\notin\boldsymbol{p}}^{N} c_i^{[\boldsymbol{p}]} \phi_{\varepsilon,i}(\boldsymbol{x}).$$

The coefficients $\boldsymbol{c}^{[\boldsymbol{p}]} = \left(c_i^{[\boldsymbol{p}]}\right)_{i\notin\boldsymbol{p}}$ are determined by solving the linear system

$$A_\varepsilon^{\boldsymbol{p},\boldsymbol{p}} \boldsymbol{c}^{[\boldsymbol{p}]} = \boldsymbol{f}^{\boldsymbol{p}}, \tag{4}$$

where $A_\varepsilon^{\boldsymbol{p},\boldsymbol{p}} = (A_\varepsilon)_{i,j}$ with $i, j \notin \boldsymbol{p}$, and $\boldsymbol{f}^{\boldsymbol{p}} = (f_i)_{i\notin\boldsymbol{p}}$. Note that $\boldsymbol{c}^{\boldsymbol{p}} = (c_i)_{i\notin\boldsymbol{p}}$ and $\boldsymbol{c}_{\boldsymbol{p}} = (c_i)_{i\in\boldsymbol{p}}$ are subvectors of $\boldsymbol{c}$, while $\boldsymbol{c}^{[\boldsymbol{p}]}$ is the solution to the above system. Importantly, $\boldsymbol{c}^{[\boldsymbol{p}]} \neq \boldsymbol{c}^{\boldsymbol{p}}$ in general.

By means of Eq. 3 instead of Eq. 4, the points of the validation set $\mathcal{V}$ are used to compute the errors

$$\boldsymbol{e}_{\boldsymbol{p}} := \boldsymbol{e}_{\boldsymbol{p}}(\varepsilon) = \boldsymbol{f}_{\boldsymbol{p}} - q^{[\boldsymbol{p}]}(f, \boldsymbol{x}_{\boldsymbol{p}}) = (f_{p_1} - q^{[\boldsymbol{p}]}(f, \boldsymbol{x}_{p_1}), \ldots, f_{p_v} - q^{[\boldsymbol{p}]}(f, \boldsymbol{x}_{p_v}))^T. \tag{5}$$

If A_ε and $\boldsymbol{c}$ are as in Eq. 3, the error vector $\boldsymbol{e_p}(\varepsilon)$ in Eq. 5 is uniquely determined by solving the system

$$(A_\varepsilon^{-1})_{\boldsymbol{p},\boldsymbol{p}}\boldsymbol{e_p} = \boldsymbol{c_p}, \quad \boldsymbol{e_p} \in \mathbb{R}^v,$$

where $(A_\varepsilon^{-1})_{\boldsymbol{p},\boldsymbol{p}} = (A_\varepsilon^{-1})_{i,j}$ for $i, j \in \boldsymbol{p}$, and $\boldsymbol{c_p} = (c_i)_{i\in\boldsymbol{p}}$ [17]. The vectorized index $\boldsymbol{p}$ extracts rows and columns corresponding to $i, j \in \boldsymbol{p}$ from the original matrix.

By concatenating the validation error vectors across all k folds, we define the error vector

$$\boldsymbol{e}(\varepsilon) = (\boldsymbol{e}_{\boldsymbol{p}_1}^T, \ldots, \boldsymbol{e}_{\boldsymbol{p}_k}^T)^T(\varepsilon).$$

In particular, setting $\boldsymbol{p} = p \in \{1, \ldots, N\}$ in Eqs. 4 and 5, the LOOCV scheme is obtained, and the optimal value of ε^* is found by minimizing the error function:

$$\text{LOOCV}(\varepsilon) = ||e_p(\varepsilon)||_\infty = \max_{p=1,\ldots,N} \left| \frac{c_p}{(A_\varepsilon^{-1})_{p,p}} \right|.$$

The reader can find a comprehensive treatment with examples of the drawbacks related to the search of the shape parameter in [11, Chapter 17].

3 Genetic Algorithm

In the field of evolutionary computation [15], GAs stand out for their ease of use and effectiveness. They are a kind of optimization process that emulates natural selection and reproduction to solve an optimization problem [14]. Like in biology, causality plays a key role and can be tweaked according to the needs. They owe their popularity to the fact that they are more powerful than a generic random search algorithm and faster than an exhaustive search algorithm while requiring no additional information on continuity or derivatives. In what follows, we describe the key components of GAs:

Fitness Function The function that the algorithm attempts to optimize. It has to be chosen carefully because it influences all the search process. It quantifies how good a solution is, and it is the only way the user has to express information about the solution.

Chromosome It is the number representation (in this paper, binary representation is considered only) of the candidate solution of the problem encoded as a vector of parameters (genes).

Population It is the set of all existing chromosomes.

Selection The selection criterion consists of choosing the two chromosomes with the highest fitness (see [14] for other selection methods).

Crossover The crossover operator resembles the crossing-over phases during cellular meiosis. It consists of taking the two best chromosomes (parents) from the selection and creating two new chromosomes (offspring) by swapping elements.

Mutation The mutation operation consists of randomly selecting one or more bits in the crossovered chromosomes and changing them with a certain probability.

The optimization process begins with defining a fitness function. Next, a random population of fixed size is generated and encoded as chromosomes. The population size remains constant throughout the optimization. A maximum number of iterations is defined as the stopping criterion. During each iteration, every chromosome in the population is evaluated using the fitness function. The two best-performing chromosomes are then selected based on their fitness. These are used in crossover and mutation operations to create new offspring. The newly generated chromosomes are incorporated into the population, which is reevaluated using the fitness function. The population is then adjusted back to its original size by retaining the best individuals. This process is repeated until the maximum number of iterations is reached. The solution of the optimization process is the chromosome in the population with the highest fitness value. Figure 1 displays the steps followed by the GA.

4 Algorithms

This section provides a detailed exploration of how genetic optimization works when applied to an interpolation task. Let us assume a set of points X is given along with their corresponding set of data values F. To implement genetic optimization (refer to Algorithm 1) and measure the validation error during the optimization process, the initial step involves splitting the sets X and F into $X_{train}, X_{val}, F_{train}, F_{val}$ into training and validation subsets $X_{train}, X_{val}, F_{train}, F_{val}$ ensuring the following proportions:

$$|X_{train}| = \lfloor 0.8 \times |X| \rfloor, \qquad |X_{val}| = \lceil 0.2 \times |X| \rceil,$$
$$|F_{train}| = \lfloor 0.8 \times |F| \rfloor, \qquad |F_{val}| = \lceil 0.2 \times |F| \rceil.$$

Algorithm 1 forms the foundation of the optimization process. It begins by initializing a population of fixed size and iteratively performs fitness evaluation, crossover, and mutation operations over a specified number of iterations. After each iteration, the population is trimmed back to its original size. Once the process concludes, the algorithm outputs the optimal shape parameter for the interpolation task.

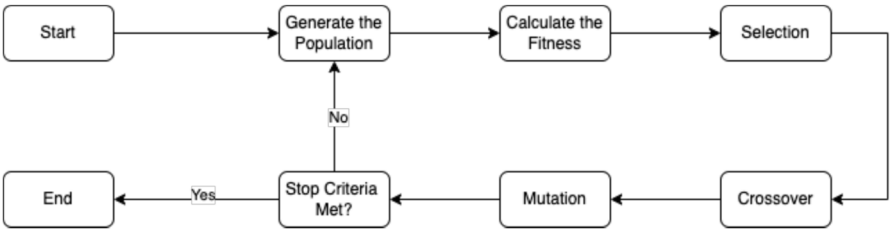

Fig. 1 Genetic algorithm flowchart

Algorithm 1 GA$(X_{train}, F_{train}, X_{val}, \phi, g, niter, sol\_per\_pop)$.

Input: X_{train}: data points, F_{train}: data values, X_{val}: evaluation points, ϕ: RBF function, g: fitness function, $niter$: number of generations, sol_per_pop: population cardinality.

Output: ε^*: best shape parameter.

$\boldsymbol{\varepsilon} \rightarrow$ generate a random population of cardinality sol_per_pop
for $i = 1 : niter$ **do**
  $\mathbf{q_f}_{\varepsilon_j} \rightarrow$ **RBF**$(X_{train}, \tilde{X}_{train}, F_{train}, X_{val}, \varepsilon_j, \phi)$, $\quad j = 1, \ldots, nstart$ (call to **Algorithm** 2)
  $\boldsymbol{g} \rightarrow (g(\mathbf{q_f}_{\varepsilon_1}), \ldots, g(\mathbf{q_f}_{\varepsilon_{sol\_per\_pop}}))$
  $\varepsilon_{p_1}, \varepsilon_{p_2} \rightarrow$ argmax $\boldsymbol{g}$
  $\varepsilon_{c_1}, \varepsilon_{c_2} \rightarrow$ **CROSSOVER**$(\varepsilon_{p_1}, \varepsilon_{p_2})$ (call to **Algorithm** 3)
  perform mutation on $\varepsilon_{c_1}, \varepsilon_{c_2}$
  supersede argmin $\boldsymbol{g}$ with $\varepsilon_{c_1}, \varepsilon_{c_2}$
end for
$\varepsilon^* \rightarrow$ argmax $\boldsymbol{g}$

To evaluate the quality of the approximant during the optimization, we introduce the maximum absolute error (MAE) on the validation set, defined as follows:

$$\mathrm{MAE}_{X_{val}, F_{val}}(\mathbf{q_f}) = \max_{\boldsymbol{x}_i \in X_{val},\, f_i \in F_{val}} |q(f, \boldsymbol{x}_i) - f_i|, \tag{6}$$

where X_{val} and F_{val} denote the data points in the validation set and their corresponding values, respectively. The vector $\mathbf{q_f} = (q(f, \boldsymbol{x}_1), \ldots, q(f, \boldsymbol{x}_{k_{val}}))$, represents the approximations at the validation points, with $k_{val} = |X_{val}|$. In this framework, the fitness function g is defined as the reciprocal of the $\mathrm{MAE}_{X_{val}, F_{val}}(\cdot)$ computed between the known values F_{val} and the approximations generated by Algorithm 2, which performs the interpolation process, at the points X_{val} or a given value of ε.

Algorithm 2 RBF$(X, \tilde{X}, F, \bar{X}, \varepsilon, \phi)$.

Input: X: data points, F: data values, $\bar{X}$: evaluation points, ε: shape parameter, ϕ: RBF function.

Output: $\mathbf{q_f}$: evaluation of the approximated solution on $\bar{X}$.

Solve the interpolation system Eq. 3

Algorithm 3 presents a pseudocode representation of the crossover operation. It swaps half of the elements between two input vectors, producing two new vectors, each containing half of the elements from the inputs.

Algorithm 3 CROSSOVER(v_1, v_2).

Input: v_1: vector, v_2: vector.

Output: v_1, v_2: crossovered version of the input vectors.

$l \rightarrow length(v_1)$
$t \rightarrow v_1[l/2:]$
$v_1[l/2:] \rightarrow v_2[l/2:]$
$v_2[l/2:] \rightarrow t$

Algorithm 4 encloses the entire process, from searching for the optimal shape parameter to evaluating the interpolant on the set $\bar{X}$. It begins by dividing the data

into training and validation sets, defining the fitness function, and performing the optimization using Algorithm 1. Once the optimal value of ε^* is identified, it computes the approximate solution on $\bar{X}$ by constructing an RBF interpolant by Algorithm 2.

Algorithm 4 RBF-GA$(X, F, \bar{X}, g, \phi, niter, sol\_per\_pop)$.

Input: X: data points, F: data values, $\bar{X}$: evaluation points, g: fitness function, ϕ: RBF function, $niter$: number of generations, sol_per_pop: population cardinality.

Output: $\mathbf{q_f}$: evaluation of the interpolated solution on $\bar{X}$.

Split X and F in $X_{train}, X_{val}, F_{train}, F_{val}$
Set $g \rightarrow \frac{1}{\overline{\text{MAE}}_{X_{val}, F_{val}}(\cdot)}$
$\varepsilon^* \rightarrow$ **GA**$(X_{train}, F_{train}, X_{val}, g, \phi, niter, sol\_per\_pop)$ (call to **Algorithm** 1)
$\mathbf{q_f} \rightarrow$ **RBF**$(X, \bar{X}, F, \bar{X}, \varepsilon^*, \phi)$ (call to **Algorithm** 2)

5 Numerical Experiments

In this section, we describe the settings for numerical experiments and discuss results. We search for the shape parameter ε in the interval $[0, \varepsilon_{max}] = [0, 20]$. We confine our treatise to the performance comparison of the GA with the LOOCV technique. When LOOCV is applied as a comparison term with GA, we evaluate the error on 500 equally spaced points in the search interval. In the case of GA, we consider all the numbers in the search interval with 3 decimal digits, multiplying them by 1000 to have integers. We transform them into 15-digit binary (0-padding short numbers) to obtain the chromosomes. The fitness function used is the reciprocal of the MAE Eq. 6, as shown in Algorithm 4.

LOOCV estimates the error for individual data points, while GA assesses the error across a collection of points. To ensure a fair comparison, we utilize three distinct datasets, training, validation, and test sets, and integrate them in the following manner. For LOOCV, the dataset X consists of the combined training and validation sets. In contrast, for GA, X represents the training set, while the validation set is employed to calculate the error. Once the optimal value of the shape parameter ε^* is found for both approaches, an RBF interpolant is constructed for each method using the union of the training and validation sets. Finally, the error is measured on the test set.

We perform the experiments on quasi-random Halton data points and random points in the domain $\Omega = [0, 1]^2$. We consider the following three RBFs

$$\phi(\varepsilon r) = \begin{cases} \exp(-\varepsilon r)(\varepsilon r + 1), & \text{Matérn } C^2 \quad \text{(M2)} \\ \max(1 - \varepsilon r, 0)^4 (4\varepsilon r + 1), & \text{Wendland } C^2 \quad \text{(W2)} \\ \exp(-\varepsilon^2 r^2), & \text{Gaussian } C^\infty \quad \text{(G)} \end{cases}$$

and the test functions:

$$f_1(\boldsymbol{x}) = 0.75 \exp\left[-\frac{(9x_1 - 2)^2}{4} - \frac{(9x_2 - 2)^2}{4}\right] + 0.75 \exp\left[-\frac{(9x_1 - 2)^2}{49} - \frac{9x_2 + 1}{10}\right] +$$

Table 1 Numerical results achieved for M2, W2, and G kernels using GA and LOOCV using the test function f_1 on Halton points

N	Method	M2			W2			G		
		Time (s)	$e(f_1)$	ε^*	Time (s)	$e(f_1)$	ε^*	Time (s)	$e(f_1)$	ε^*
200	GA	0.83	9.29e−03	3.96	0.58	8.31e−03	0.17	0.42	9.02e−03	5.44
	LOOCV	1.55	1.01e−02	1.00	1.55	1.19e−02	1.00	1.23	9.14e−03	5.28
400	GA	1.21	3.84e−03	0.18	2.20	3.71e−03	0.17	0.94	5.59e−04	6.14
	LOOCV	6.11	3.59e−03	0.56	6.11	3.97e−03	0.56	4.74	2.65e−04	5.84
800	GA	2.74	6.55e−04	1.02	3.79	1.03e−03	0.01	3.54	2.33e−05	5.63
	LOOCV	34.2	9.97e−04	0.96	34.2	1.11e−03	0.96	32.3	1.64e−05	6.16
1600	GA	44.5	4.75e−04	1.61	33.9	7.94e−04	1.02	37.2	3.12e−06	6.98
	LOOCV	186	4.87e−04	0.68	186	5.73e−04	0.68	167	1.84e−06	6.52

$$+0.5\exp\left[-\frac{(9x_1-7)^2}{4}-\frac{(9x_2-3)^2}{4}\right]-0.2\exp\left[-(9x_1-4)^2-(9x_2-7)^2\right],$$

$$f_2(\boldsymbol{x})=\frac{(64-81((x_1-0.5)^2+(x_2-0.5)^2))^{\frac{1}{2}}}{9}-0.5.$$

All the above kernels are SPD: M2 and G are globally supported, while W2 is compactly supported.

To evaluate the algorithm's performance, we compute the execution time in seconds and the error

$$e(f):=\max_{\boldsymbol{x}\in\Omega}|f(\boldsymbol{x})-q(f,\boldsymbol{x})|.$$

The tests are performed using Python 3.9.12 on a 1.2 GHz Quad-Core Intel Core i7 processor, 16 GB 3733 MHz LPDDR4X RAM MacBook Air (2020).

We perform two different performance comparisons. In the first case study, we consider four different dataset sizes N for each type of data points and compare the

Table 2 Numerical results achieved for M2, W2, and G kernels using GA and LOOCV using the test function f_1 on random points

N	Method	M2			W2			G		
		Time (s)	$e(f_1)$	ε^*	Time (s)	$e(f_1)$	ε^*	Time (s)	$e(f_1)$	ε^*
200	GA	0.32	1.03e−02	4.10	0.97	9.48e−03	0.35	0.63	7.24e−01	4.18
	LOOCV	2.09	9.13e−03	2.12	2.69	9.40e−03	0.36	1.89	4.65e−02	5.48
400	GA	1.11	1.53e−02	1.54	1.23	2.67e−02	1.02	1.41	2.12e−03	5.12
	LOOCV	7.05	1.71e−02	3.44	7.61	2.25e−02	0.88	6.83	5.07e−03	5.60
800	GA	5.36	5.37e−03	6.14	5.88	5.88e−03	1.31	6.26	2.16e−04	5.73
	LOOCV	36.9	1.36e−03	2.28	41.3	1.60e−03	0.52	38.0	1.89e−05	6.60
1600	GA	39.3	5.98e−04	0.29	39.2	7.13e−04	0.02	49.7	1.80e−05	8.19
	LOOCV	198	5.93e−04	0.72	211	5.95e−04	0.16	201	1.27e−06	6.08

Table 3 Numerical results achieved for M2, W2, and G kernels with GA and LOOCV using the test function f_2 on Halton points

		M2			W2			G		
N	Method	Time (s)	$e(f_2)$	ε^*	Time (s)	$e(f_2)$	ε^*	Time (s)	$e(f_2)$	ε^*
200	GA	0.22	4.20e−03	0.01	0.22	4.21e−03	0.01	0.22	1.16e−03	0.51
	LOOCV	0.95	4.30e−03	0.04	1.20	4.54e−03	0.04	1.15	8.19e−05	2.96
400	GA	0.45	2.04e−03	0.06	0.82	2.03e−03	0.02	1.39	1.82e−05	2.47
	LOOCV	4.73	2.01e−03	0.04	6.85	2.12e−03	0.04	5.48	4.83e−05	2.76
800	GA	3.67	5.47e−04	0.51	8.16	4.51e−04	0.03	5.15	4.06e−06	3.28
	LOOCV	30.2	4.33e−04	0.04	35.3	4.58e−04	0.04	30.1	1.50e−06	4.00
1600	GA	44.2	1.52e−04	0.13	40.9	1.81e−04	0.02	28.7	2.03e−06	4.23
	LOOCV	169	1.45e−04	0.04	185	1.53e−04	0.04	167	6.62e−07	7.76

performance of GA and LOOCV within the RBF interpolation framework. In the second case, we fix the dataset size N for each type of data points and assess the accuracy of the GA approach by comparing it with a similarly fast method, namely Bayesian Optimization (BO).

BO is a statistical technique widely used in machine learning for the optimization of black-box or computationally expensive functions. In the context of hyperparameter tuning, it allows one to avoid the computation and evaluation of the approximant for parameter values that are far from optimal, leading to a significant reduction in computational cost during the exploration of the parameter domain. BO operates by constructing a probabilistic surrogate model of the error function, typically based on a Gaussian process, which is iteratively updated as new evaluations are performed. At each iteration, an acquisition function is employed to select the next evaluation point, chosen as the maximizer of the acquisition function. The corresponding objective function evaluation is then used to update the surrogate model. The updated distribution is subsequently exploited to guide the selection of new sampling points in the parameter space X. For further details on BO, we refer the reader to [4, 23].

Table 4 Numerical results achieved for M2, W2, and G kernels with GA and LOOCV using the test function f_2 on random points

		M2			W2			G		
N	Method	Time (s)	$e(f_2)$	ε^*	Time (s)	$e(f_2)$	ε^*	Time (s)	$e(f_2)$	ε^*
200	GA	0.37	2.68e−02	0.51	0.30	2.23e−02	0.03	0.60	2.67e−04	1.71
	LOOCV	2.18	2.13e−02	0.04	2.27	2.26e−02	0.04	2.38	5.52e−04	2.64
400	GA	1.09	4.19e−03	0.03	1.12	4.69e−03	0.06	1.48	1.39e−04	2.51
	LOOCV	7.17	4.22e−03	0.04	7.96	4.48e−03	0.04	7.15	1.44e−04	4.24
800	GA	5.35	3.00e−03	0.26	4.67	2.91e−03	0.02	5.57	4.33e−05	2.96
	LOOCV	39.1	2.68e−03	0.04	41.1	2.84e−03	0.04	36.8	3.21e−06	4.00
1600	GA	39.9	1.10e−03	0.51	31.5	1.02e−03	0.02	34.9	1.85e−05	3.34
	LOOCV	202	8.80e−04	0.04	211	9.28e−04	0.04	194	2.33e−06	7.16

Table 5 Numerical results for M2, W2, and G kernels using GA, BO, and LOOCV on $N = 1000$ Halton points for the test functions f_1 and f_2

		M2			W2			G		
	Method	Time (s)	$e(f_i)$	ε^*	Time (s)	$e(f_i)$	ε^*	Time (s)	$e(f_i)$	ε^*
f_1	GA	7.41	1.59e−04	1.415	6.43	7.43e−04	0.231	6.27	1.28e−05	6.214
	BO	6.20	7.68e−04	1.329	4.12	7.93e−04	0.277	4.29	9.71e−06	7.491
	LOOCV	60.9	7.69e−04	1.240	58.6	7.91e−04	0.240	66.0	1.29e−05	6.440
f_2	GA	6.36	2.56e−04	0.064	6.10	2.54e−04	0.016	7.36	1.15e−06	4.168
	BO	3.37	4.11e−04	0.935	4.20	2.55e−04	0.001	3.87	1.79e−06	3.121
	LOOCV	52.3	2.61e−04	0.040	59.6	2.76e−04	0.040	50.7	1.44e−06	4.320

The results of the first case study are reported in Tables 1, 2, 3, and 4, while Tables 5 and 6 summarize the outcomes of the second performance test. Regarding BO, the acquisition function adopted in this study is the *Expected Improvement*, which accounts not only for the probability that a candidate point improves upon the current optimum, but also for the magnitude of the expected improvement. Overall, when comparing GA, LOOCV, and BO within the RBF interpolation framework, the resulting errors are generally of the same order of magnitude. A key distinction among the considered methods, however, lies in their computational cost: GA and BO are consistently faster than LOOCV, even though LOOCV evaluates the error at a single point, whereas GA and BO evaluate the error over a set of points.

Finally, in some cases, the tables show that the optimal value of the shape parameter ε^* depends on the method employed. This behavior can be attributed to the multimodal nature of the underlying optimization problem, which may admit multiple optimal or near-optimal solutions. Multimodality is common in real-world applications, where the goal is often to identify robust solutions within a search space containing several good candidates. Although, to the best of our knowledge, a general and comprehensive theoretical framework for assessing the robustness of GA is still lacking, some steps in this direction have been taken (see, e.g., [1, 14]). We therefore empirically evaluated the robustness of the algorithm by perturbing the training and validation sets, obtaining comparable results in all tested cases and thus indicating stable performance under such perturbations.

Table 6 Numerical results for M2, W2, and G kernels using GA, BO, and LOOCV on $N = 1000$ random points for the test functions f_1 and f_2

		M2			W2			G		
	Method	Time (s)	$e(f_i)$	ε^*	Time (s)	$e(f_i)$	ε^*	Time (s)	$e(f_i)$	ε^*
f_1	GA	5.81	6.70e−04	1.349	7.00	6.14e−04	0.248	6.47	2.02e−05	6.346
	BO	4.33	6.63e−04	0.788	6.07	6.85e−04	0.001	4.33	1.45e−04	6.465
	LOOCV	68.7	6.76e−04	1.520	93.7	6.77e−04	0.280	71.2	8.53e−05	7.040
f_2	GA	6.77	1.85e−03	0.448	11.3	8.70e−04	0.016	7.57	1.03e−05	4.538
	BO	3.70	1.08e−03	0.475	6.59	8.30e−04	0.001	5.20	2.07e−05	4.783
	LOOCV	70.5	8.60e−04	0.040	77.7	9.13e−04	0.040	79.0	2.66e−05	6.000

6 Conclusions and Future Work

In this paper, we employed the GA to search for the optimal shape parameter in RBF interpolation. This approach proved to be a valuable alternative to CV techniques, as it significantly reduces the computational effort involved in the parameter search process. From the numerical evidence of the tests presented in this work, a comparison of parameter search for interpolation using LOOCV, BO, and GA reveals that the resulting error values are typically of similar magnitude. However, the key difference lies in computational efficiency: the GA approach often achieves results in significantly less time, with calculation times reduced to nearly half in most cases. This makes the GA particularly suitable for large-scale problems or scenarios where computational resources are limited.

A potential extension of this work could involve improving the performance of the GA to achieve higher accuracy with respect to the other examined approaches. This could be accomplished by applying GA optimization to the Partition of Unity framework [26], enabling the local determination of shape parameters for each subdomain, as done in [6]. Moreover, GA could also be used in several applications of RBFs, for instance, in numerical integration [9], approximation of signal on graphs [22], numerical treatment of Fredholm integral equations [5, 7], partial differential equations [25], or real-world applications [24].

Acknowledgements The authors are grateful to the anonymous reviewers for carefully reading the manuscript and for their precise and helpful suggestions which allowed to improve the work.

Funding Open access funding provided by Università degli Studi della Basilicata within the CRUI-CARE Agreement. This work has been supported by the Gruppo Nazionale Calcolo Scientifico-Istituto Nazionale di Alta Matematica (GNCS-INdAM) as part of the GNCS-INdAM 2025 project "Metodi di approssimazione globale per operatori integrali e applicazioni alle equazioni funzionali."It has been also supported by the Spoke 1 "FutureHPC & BigData"of ICSC - Centro Nazionale di Ricerca in High-Performance Computing, Big Data and Quantum Computing, funded by European Union - NextGenerationEU and by the PRIN 2022 PNRR no. P20229RMLB financed by the European Union - NextGeneration EU and the Italian Ministry of University and Research (MUR). Moreover, the work has been supported by the Fondazione CRT, project 2022 "Modelli matematici e algoritmi predittivi di intelligenza artificiale per la mobilità sostenibile." This research has been accomplished within the RITA "Research ITalian network on Approximation,"the UMI Group TAA "Approximation Theory and Applications,"and the SIMAI Activity Group ANA&A "Numerical and Analytical Approximation of Data and Functions with Applications."

Data Availability No datasets were generated or analysed during the current study.

Declarations

Conflict of Interest The authors declare no competing interests.

References

1. Brizuela CA, Sannomiya N (2001) Robustness and diversity in genetic algorithms for a complex combinatorial optimization problem. Int J Syst Sci 32(9):1161–1168
2. Buhmann MD (2003) Radial basis functions: theory and implementation. Cambridge Monographs on Applied and Computational Mathematics, vol 12. Cambridge University Press, Cambridge
3. Cavoretto R, De Rossi A, Haider A, Lancellotti S (2024) Comparing deterministic and statistical optimization techniques for the shape parameter selection in RBF interpolation. Dolomites Res Notes Approx 17:48–55
4. Cavoretto R, De Rossi A, Lancellotti S (2024) Bayesian approach for radial kernel parameter tuning. J Comput Appl Math 441:115716
5. Cavoretto R, De Rossi A, Laguardia AL, Mezzanotte D, Occorsio D, Russo MG (2026) An RBF-based Nyström method for Second–Kind Fredholm integral equations. J Comput Appl Math 474:116968 https://doi.org/10.1016/j.cam.2025.116968
6. Cavoretto R, De Rossi A, Lancellotti S, Romaniello F (2024) Parameter tuning in the radial kernel-based partition of unity method by Bayesian optimization. J Comput Appl Math 451:116108
7. Cavoretto R, De Rossi A, Mezzanotte D (2024) A review of radial kernel methods for the resolution of Fredholm integral equations of the second kind. Constr Math Anal 7:142–153. https://doi.org/10.33205/cma.1538581
8. Cavoretto R, De Rossi A, Mukhametzhanov MS, Sergeyev YD (2021) On the search of the shape parameter in radial basis functions using univariate global optimization methods. J Global Optim 79:305–327
9. Cavoretto R, De Rossi A, Sommariva A, Vianello M (2022) RBFCUB: a numerical package for near-optimal meshless cubature on general polygons. Appl Math Lett 125:107704
10. Celisse A, Robin S (2008) Nonparametric density estimation by exact leave-p-out cross-validation. Comput Statist Data Anal 52:2350–2368
11. Fasshauer GE (2007) Meshfree approximation methods with MATLAB. World Scientific, Singapore
12. Fasshauer G, McCourt M (2015) Kernel-based approximation methods using MATLAB. Interdiscip Math Sci, vol 19. World Scientific, Singapore
13. Golbabai A, Mohebianfar E, Rabiei H (2015) On the new variable shape parameter strategies for radial basis functions. Comp Appl Math 34:691–704
14. Goldberg DE (1989) Genetic algorithms in search. Optimization and Machine Learning. Addison-Wesley Longman Publishing Co., Inc, USA
15. Kinnear KE (1994) Advances in genetic programming. MIT Press, Cambridge, MA, USA
16. Ling L, Marchetti F (2022) A stochastic extended Rippa's algorithm for LpOCV. Appl Math Lett 129:107955
17. Marchetti F (2021) The extension of Rippa's algorithm beyond LOOCV. Appl Math Lett 120:107262
18. Noorizadegan A, Chen C-S, Cavoretto R, De Rossi A (2024) Efficient truncated randomized SVD for mesh-free kernel methods. Comput Math Appl 164:12–20
19. Rippa S (1999) An algorithm for selecting a good value for the parameter c in radial basis function interpolation. Adv Comput Math 11:193–210
20. Schaback R (1999) Native Hilbert spaces for radial basis functions I. In: New Developments in Approximation Theory, pp 255–282. Birkhäuser Basel, Basel
21. Scheuerer M (2011) An alternative procedure for selecting a good value for the parameter c in RBF-interpolation. Adv Comput Math 34:105–126
22. Stankovic L, Dakovic M, Sejdic E (2019) Introduction to graph signal processing, pp 3–108. Springer, Cham. https://doi.org/10.1007/978-3-030-03574-7_1
23. Snoek J, Larochelle H, Adams RP (2012) Practical Bayesian optimization of machine learning algorithms. Adv Neural Inf Process Syst 25:2960–2968
24. Silvestrin LP, van Zanten H, Hoogendoorn M, Koole G (2023) Transfer learning across datasets with different input dimensions: an algorithm and analysis for the linear regression case. J Comput Math Data Sci 9:100086. https://doi.org/10.1016/j.jcmds.2023.100086
25. Wang F, Ahmad I, Ahmad H, Alsulami MD, Alimgeer KS, Cesarano C, Nofal TA (2021) Meshless method based on RBFs for solving three-dimensional multi-term time fractional PDEs arising in engineering phenomenons. J King Saud Univer - Sci 33(8):101604
26. Wendland H (2002) Fast evaluation of radial basis functions: methods based on partition of unity. In: Approximation Theory X: Wavelets, Splines, and Applications, pp 473–483

27. Wendland H (2005) Scattered data approximation. cambridge monographs on applied and computational mathematics, vol 17. Cambridge University Press, Cambridge